On the Trust of Trusted Computing in the Post-Snowden Age

Feng Hao

School of Computing Science
Newcastle University, UK

Workshop on Analysis of Security APIs
13 July, 2015

Acknowledgment

Based (partly) on the following paper

- Feng Hao, Dylan Clarke, Avelino Zorzo, "Deleting Secret Data with Public Verifiability," accepted by IEEE Transactions on Dependable and Secure Computing, 2015, in press.

Trusted Computing

- A broad term to refer to hardware-based security solutions
- Mainly driven by IT companies
  - 1999, started by Trusted Computing Platform Alliance (Compaq, HP, IBM, Intel and Microsoft)
  - 2003, succeeded by Trusted Computing Group (AMD, HP, IBM, Intel, Microsoft, etc)
- Initial focus on PCs
- Now increasing emphasis on mobile devices

The Difference Between Being 'Trusted' and 'Trustworthy'

"Rupert Murdoch ... referred consistently to his pride in the Sun as 'a trusted news source.' Trusted is the word he used, not trustworthy. We know the Sun is not trustworthy and so does he. He uses the word 'trusted' deliberately. ***Hitler was trusted, it transpired he was not trustworthy.*** *He also said of ..."*

– Russell Brand, in the Guardian, on Murdoch, the Sun, and the miserable state of the news industry

Different terminologies

- Microsoft renamed “Trusted Computing” to “Trustworthy Computing” (TC)
- But IBM and Intel stick to “Trusted Computing” (TC)
- Free Software Foundation calls “Treacherous Computing” (TC)
- In the end, the term “**Trusted Computing**” sinks in and is widely used

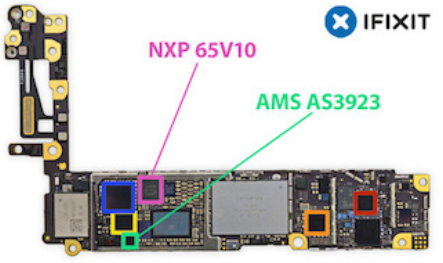
Controversies around Trusted Computing

- Many arguments against TC
  - Users lose control on privacy
  - Customer lock-in
  - Hinder free and open source software movement
- Like any other technology, TC can be used for good and bad
- In this talk, we focus only on technical aspects of TC

Essence of Trusted Computing

- Tamper-resistance
  - Tamper-resistant hardware with embedded CPU, secure memory
- Secure key storage
  - Secret keys always kept inside secure memory
- Key management
  - Via a set of APIs (most difficult part in the design)

Embodiments of Trusted Computing



- Trusted Computing Base (TCB), Trusted Platform Module (TPM), Hardware Security Module (HSM), Smart Card (contact/contactless, USB), Secure Element

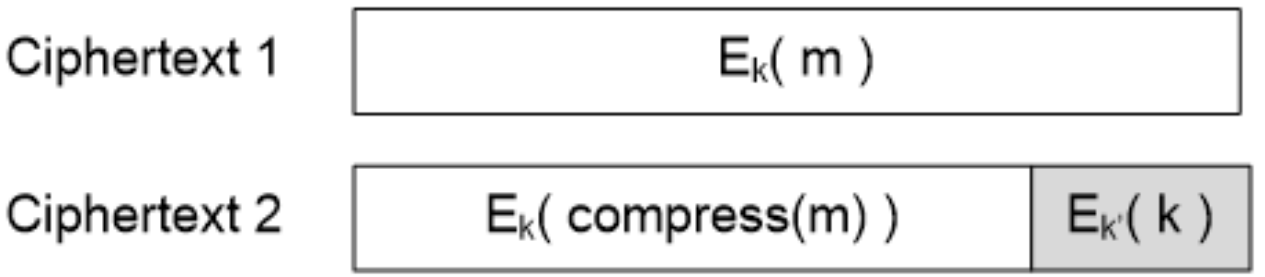
A simple API example

- Encryption
  - Host $\rightarrow$TPM: m
  - TPM $\rightarrow$Host: $c = E_k(m)$
- Decryption
  - Host $\rightarrow$TPM: $c = E_k(m)$
  - TPM $\rightarrow$Host: m

How to ensure if the implementation is correct?

- The industrial practice
  - Unit testing
  - Test vectors
- The intuitive reasoning
  - If decryption always correctly recovers the original plaintext, the implementation should be correct.
- We argue this is not sufficient in face of state-funded adversaries.

A trapdoor attack

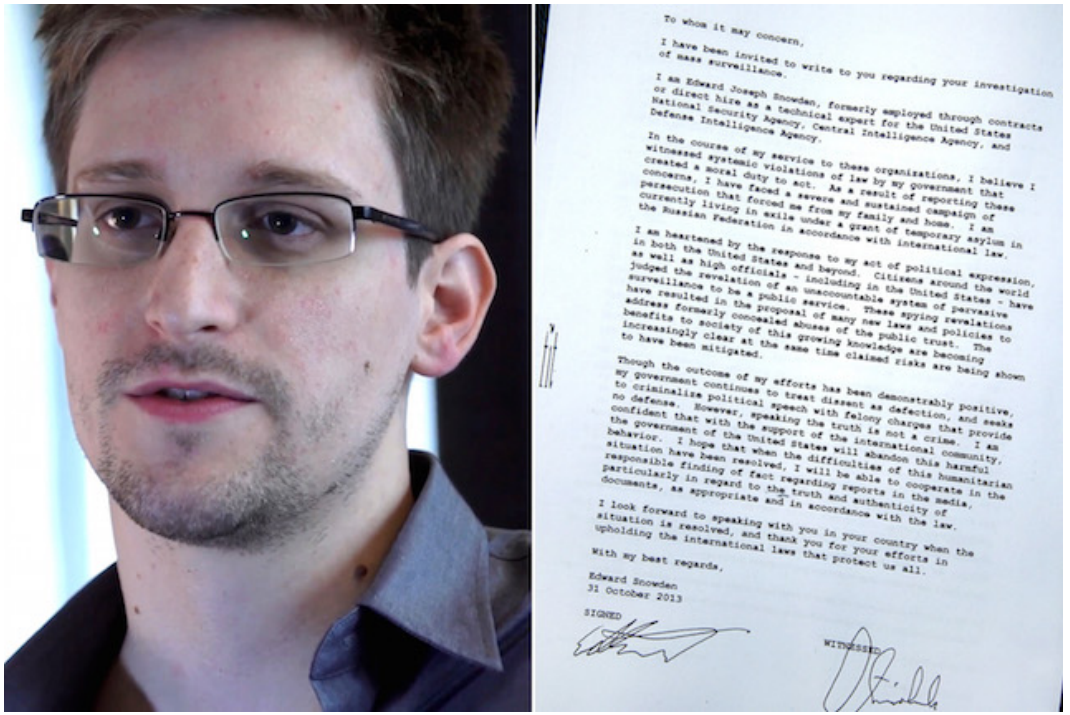


- TPM compresses the message before encryption, and adds a trapdoor block to make equal length
  - Trapdoor block is the decryption key wrapped under a trapdoor key
- Assume encryption schemes are semantically secure
  - Ciphertext is indistinguishable from random
  - The attack is undetectable

Why would any TPM manufacturer do that?

- Incentive
  - Manufacturers appears to have no incentive to compromise security of their own products
  - But a state-funded adversary may have the incentive to coerce/bribe manufacturers to do that
- Revelations from Snowden
  - NSA has been trying to insert trapdoors to security products
  - Actually, any state-funded adversary would do just the same

Trapdoor in Trusted Computing - truth or rumor?



To whom it may concern,

I have been invited to write to you regarding your investigation of mass surveillance.

I am Edward Joseph Snowden, formerly employed through contracts or direct hire as a technical expert for the United States National Security Agency, Central Intelligence Agency, and Defense Intelligence Agency.

In the course of my service to these organizations, I believe I witnessed systemic violations of law by my government that created a moral duty to act. As a result of reporting these concerns, I have faced a severe and sustained campaign of persecution that forced me from my family and home. I am currently living in exile under a grant of temporary asylum in the Russian Federation in accordance with international law.

I am heartened by the response to my act of political expression, in both the United States and beyond. Citizens around the world as well as high officials - including in the United States - have judged the revelation of an unaccountable system of pervasive surveillance to be a public service. These spying revelations have resulted in the proposal of many new laws and policies to address formerly concealed abuses of the public trust. The benefits to society of this growing knowledge are becoming increasingly clear at the same time claimed risks are being shown to have been mitigated.

Though the outcome of my efforts has been demonstrably positive, my government continues to treat dissent as defection, and seeks to criminalize political speech with felony charges that provide no defense. However, speaking the truth is not a crime. I am confident that with the support of the international community, the government of the United States will abandon this harmful behavior. I hope that when the difficulties of this humanitarian situation have been resolved, I will be able to cooperate in the responsible finding of fact regarding reports in the media, particularly in regard to the truth and authenticity of documents, as appropriate and in accordance with the law.

I look forward to speaking with you in your country when the situation is resolved, and thank you for your efforts in upholding the international laws that protect us all.

With my best regards,

Edward Snowden
31 October 2013

SIGNED

WITNESSED

- How can you possibly tell?
  - No access to internal software thanks to tamper resistance

Learning from the history

- Similar trust crisis on electronic voting
  - DRE records votes and announces the winner
  - Voters have to trust the outcome, but can't verify
  - Public outcry as no way to check internal software
- Solution: trust-but-verify
  - "Software Independence" (Ron Rivest, John Wack)
  - Verify the integrity of voting software by checking its output, not its source code
  - Underpinning 20 years research on verifiable e-voting

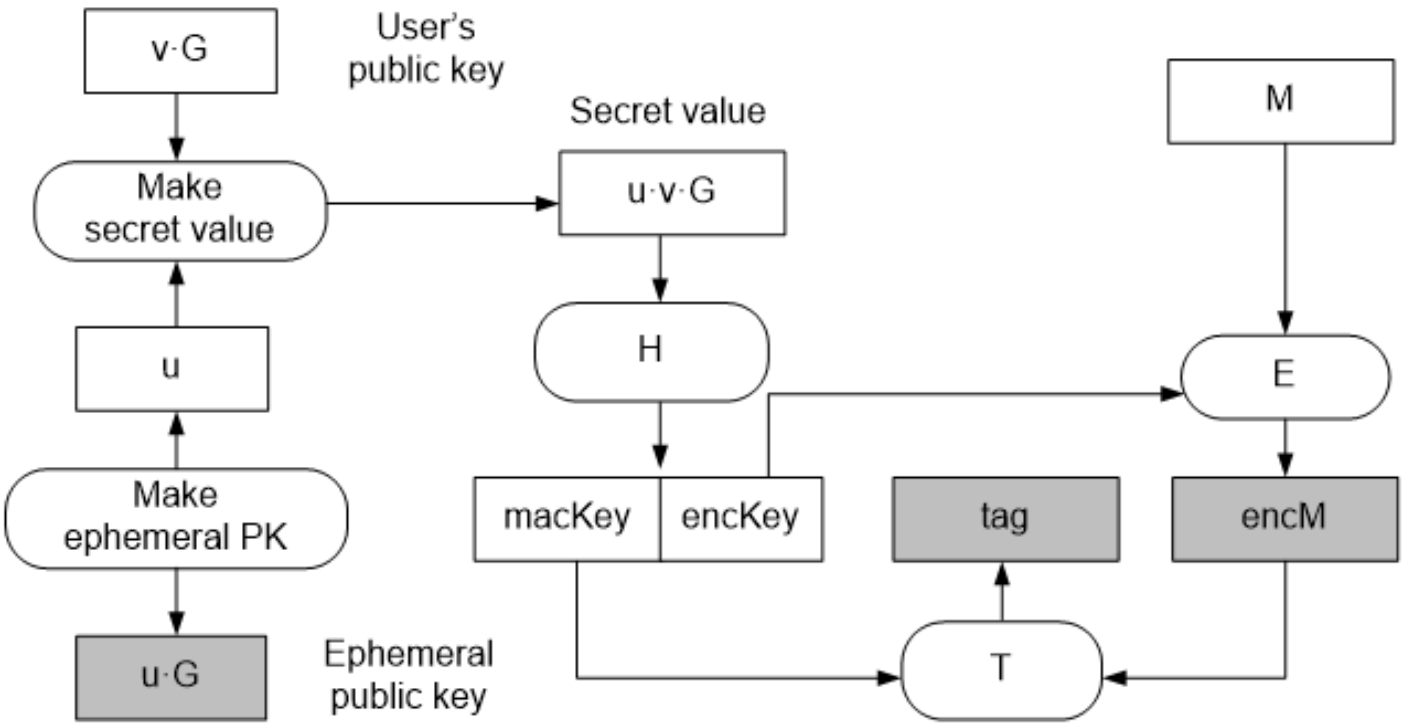
Our proposal: apply Trust-but-Verify to redesign TC API

- In the post-Snowden age
  - The threat of state-funded adversary cannot be ignored
- Apply "trust-but-verify" principle
  - Many existing APIs need to be re-designed

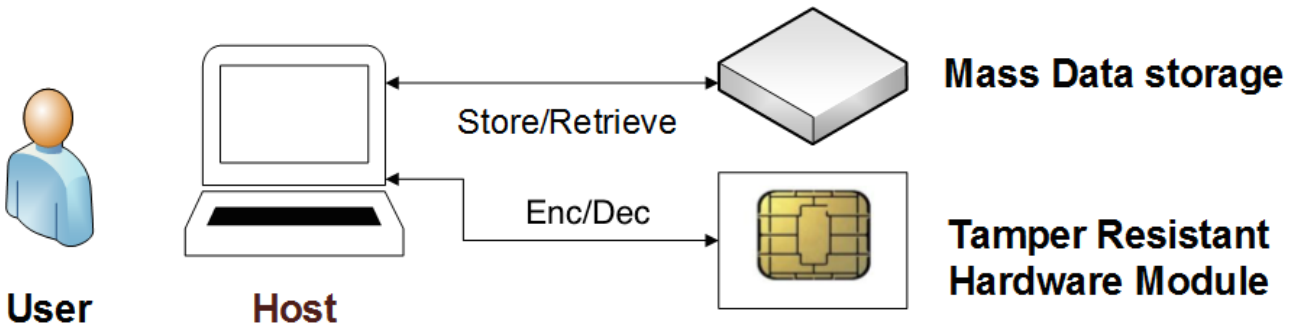
An example: data encryption

- Suppose we build TPM-based disk encryption
  - Using Diffie-Hellman Integrated Encryption Scheme (DHIES)
- DHIES
  - Abdalla, Bellare and Rogaway (CT-RSA, 2001)
  - Included in standards of ANSI X9.63, IEEE P1363a and SECG

Overview of DHIES



System architecture



- API calls
  - Key Generation
  - Encryption
  - Decryption
- Other APIs omitted (e.g., authentication, key revocation)

Key Generation

Key generation:

$$\begin{array}{lcll} \text{Host} & \rightarrow & \text{TPM} & : \quad 1^k,\ C \\ & & \text{TPM} & : \quad \text{Generate Prv}_{C_i} := d_{C_i} \\ \text{TPM} & \rightarrow & \text{Host} & : \quad \text{Pub}_{C_i} := d_{C_i} \cdot G,\ C_i \end{array}$$

Encryption/Decryption

Encryption:

$$\begin{array}{lcll} \text{Host} & \rightarrow & \text{TPM} & : \quad C_i,\ m \\ \text{TPM} & \rightarrow & \text{Host} & : \quad Q_\eta := d_\eta \cdot G,\ H(k_c),\ E^{\text{Auth}}_{k_\eta}(m) \end{array}$$

Decryption:

$$\begin{array}{lcll} \text{Host} & \rightarrow & \text{TPM} & : \quad C_i,\ Q_\eta,\ H(k_c),\ E^{\text{Auth}}_{k_\eta}(m) \\ \text{TPM} & \rightarrow & \text{Host} & : \quad m \end{array}$$

State-funded adversary

- This design is trivially subject to the trap-door attack
- A state-funded adversary is able to read all encrypted traffic

Applying trust-but-verify

Audit:

$$\begin{array}{lcll} \text{Host} & \rightarrow & \text{TPM} & : \quad C_i,\ Q_\eta,\ H(k_c) \\ \text{TPM} & \rightarrow & \text{Host} & : \quad d_{C_i} \cdot Q_\eta,\ \ldots \\ & & & \quad \text{ZKP}_\eta\left[\log_G d_{C_i} \cdot G = \log_{Q_\eta} d_{C_i} \cdot Q_\eta\right] \end{array}$$

- Explicitly verify if encryption was done properly
- Based on Chaum-Pedersen ZKP
- More details in the paper

Implementation

- Full implementation on a resource-constrained Java Card
- A non-trivial implementation challenge
- Java Card code published in public domain (see paper)

Outlook: small step, long journey

- We started a small step, but it will be a long journey.
- Many research questions remain: e.g.,
  - How to ensure if the random number is generated correctly

Summary

- Existing assumptions about Trusted Computing
  - Either complete trust or totally distrust (i.e., black/white)
  - However, neither captures realistic requirements
- We propose "trust but verify"
  - A well accepted principle in 20 years e-voting research
  - But almost entirely neglected in the field of Trusted Computing
- Future work
  - Re-design existing TPM APIs based on the new principle
  - Many open research problems, e.g., random number generation

From Trusted Computing to Trust-but-Verify Computing

Thank you!

For more technical details, see
`https://eprint.iacr.org/2014/364.pdf`